PPL – Assignment 4

Sagiv Nethanel 203308069

Roy Soldin 204542179

Part 1 - Concept Questions

1. **Find MGUs for the following pairs of type expressions**

   a) `[T1*[T1->T2]->N]`, `[[T3->T4] *[T5->Number]->N]`

   MGU: `{T1 = [T3 -> T4], T2 = Number, T5 =[T3 -> T4]}`

   b) `[T1*[T1->T2]->N]`, `[Number * [Symbol->T3]->N`

   MGU: `Not exist!`

   c) `T1`, `T2`

   MGU: `{T1 = T2}`

   d) `Number, Number`

   MGU: `{}`

2. We can typecheck letrec expressions without specific problems related to recursion and without the need for a recursive environment like we had in the interpreter because there's no need for a recursive environment typewise since a recursive function must have the same signature in every frame, hence of the same type. i.e. a recursive function is always of the type [T1*…*Tn->Tret] for some type expressions T1,…,Tn , Tret, otherwise it cannot be recursive. In contrast, when considering evaluation of expressions (value matters), one must take into account the recursive environment.

3. In the type equation implementation - we represent Type Variables (TVar) with a content field (which is a box which contains a Type Expression value or #f when empty). In this representation, we can have a TVar refer in its content to another TVar - repeatedly, leading to a chain of TVars. Design a program which, when we pass it to the type inference algorithm, creates a chain of length 4 of Tvar1 → Tvar2 → Tvar3 → Tvar4. Write a test to demonstrate this configuration.

```
(lambda (t1) (lambda(t2) (lambda(t3) (lambda(t4) (t4)))))
```